一、基础选择(每题 2 分，共 30 分)

1. 以下程序设计语言中，哪一种语言属于解释型语言?

   A. C 语言　　B. Pascal　　C. C++　　D. python

2. 以下程序设计语言中，哪一种语言的效率最高?

   A. 汇编语言　　B. C 语言　　C. python　　D. java

3. 有一个二进制数字的计算机原码为 $(100101)_{原}$，它转换成计算机补码是以下哪个(　　)

   A. $(100101)_{补}$　　B. $(111011)_{补}$　　C. $(1011010)_{补}$　　D. $(111010)_{补}$

4. 下面有关排序算法的说法中哪条是错误的(　　)

   A. 归并排序的复杂度是 $O(nlogn)$

   B. 计数排序一定比归并排序快

   C. 归并排序和快速排序的思想都是分治

   D. 归并排序的空间复杂度是 $O(n)$

5. 二维数组 A 的定义方法是 int A[5][3]，A的初始地址是 0x00 那么A[0] + 1，A + 1 的地址分别是多少(　　)

   A. 0x04,0x04　B. 0x0c,0x04　C. 0x0c,0x0c　D. 0x04,0x0c

6. C++ 的标准库 STL 中有 vector，queue，stack，set 等，如果我们想要使得存进去的数字自动排序(从小到大或者从大到小)，以下哪些 STL 是可以做到的(　　)

   A.set　priority_queue

   B.vector queue

   C.stack queue

   D.priority_queue　queue

7. 以下几种电子元件中，存取效率最高的是（　　）

   A. 寄存器　　B. 高速缓冲存储器　　C. U 盘　　D. 固态硬盘

8. 给出一个后序遍历 CDBGFEA 和一个前序遍历 ABCDEFG 下列哪一个**可能**是该二叉树的中序遍历(　　)

   A. CBDAGFE

   B. ABCDEFG

   C. CDBAGFE

   D. CBADEFG

9. 设 `A=B=true`，`C=D=false`，以下逻辑运算表达式值为假的是（ ）

   A. (¬ A∧B)∨(C∧D∨A)

   B. ¬ (((A∧B)∨C)∧D)

   C. A∧(B∨C∨D)∨D

   D. (A∧(D∨C)) ∧B

10. 与十进制数 17.5625 对应的 8 进制数是（ ）。

    A. 21.5625

    B. 21.44

    C. 21.73

D. 21.731

组合人送大家的填空题

11.有 6 个完全一样的小球，放入 3 个标号为 1 到 3 的箱子中，其中 1 号箱子至少一个小球，2 号和 3 号可以为空，不同的方案数量有（）

12.有 5 个小朋友，其中有 2 个小朋友有两个不同颜色的小球，其余小朋友都只有一个小球，小球的颜色共计 5 种，【每人手中的小球颜色都是随机的】，则至少有两个人的小球**数量和颜色**完全相同的概率是（）

13.现在从 10 个小朋友中选 3 个小朋友参加文艺汇演，其中小朋友 1 和 2 关系很好，要么两个小朋友一起参加要么都不参加，你有（）种不同的选派方案

14.现在有 5 个不同的节目要排出场顺序，其中节目 1 和节目 2 不能安排在相邻顺序出场，节目 2 和节目 3 必须安排在相邻顺序出场，则总的出场方案有（）种

15.烧烤店有三种肉串售卖，其中肉串 A 的价格是 2 元，肉串 B 是 3 元，肉串 C 是 4 元，你有 20 元，恰好将钱花光的购买方案有（）种

二、程序阅读(判断题每题 1.5 分，选择题每题 3 分，共 40 分)

程序阅读第一题(共 13 分)

```
01  #include <iostream>
02  using namespace std;
03  int main(){
04      string s;
05      int k;
06      cin >> s >> k;
07      int l = 0, r = 0, cnt = 0, ans = 0;
08      while(r < s.size()){
09          while(r < s.size() && cnt <= k){
10              r++;
11              cnt += (s[r] == 'B');
12          }
13          ans = max(ans, r - l);
14          while(l <= r && cnt > k){
15              cnt -= (s[l] == 'B');
16              l++;
17          }
18      }
19      cout << ans << endl;
20  }
```

注：本题中输入的 k 为一个大于等于 0 的整数。

判断题

1. 第 07 行改为 `int l=r=cnt=ans=0`，程序运行结果不会改变。
2. 若将第 08 行改为 while(true)，程序运行结果不会改变。
3. 如果输入的 k 大于字符串的长度，则输出的答案一定为字符串的长度。
4. 若输入的字符串中没有大写字母 B，则无论 k 的值是多少，输出的答案都为字符串的长度。

选择题

1. （4 分）若输入为 AAABAABABAA 2，则输出为

   A. 9　　B. 7　　C. 11　　D. 8

2. 若字符串长度为 n，则该程序的时间复杂度为 （　）

   A. $O(n)$　　B. $O(n^2)$　　C. $O(nk)$　　D. $O(nlogk)$

程序阅读第二题(共 12 分)

保证n < 20，此题是一个关于点与点距离的题目

```
1    #include<cstdio>
2    #include<cmath>
3    #include<cstring>
4    typedef double db;
5    db x[20],y[20],f[20][35000];
6    template<class T> T min(T a,T b) {return a<b?a:b;}
7    db dis(int a,int b) {return sqrt((x[a]-x[b])*(x[a]-x[b])+   8  (y[a]-y[b])*
(y[a]-y[b]));}
9    int main()
10    {
11        int n;scanf("%d",&n);
12        for(int i=1;i<=n;i++) scanf("%lf%lf",&x[i],&y[i]);
13        memset(f,127,sizeof(f));
14        for(int s=1;s<=(1<<n)-1;s++)
15        for(int i=1;i<=n;i++)
16        {
17            if((s&(1<<(i-1)))==0) continue;
18            if(s==(1<<(i-1))) {f[i][s]=0;continue;}
19            for(int j=1;j<=n;j++)
20            {
21              if((s&(1<<(j-1)))==0||i==j) continue;
22              f[i][s]=min(f[i][s],f[j][s-(1<<(i-1))]+dis(i,j));
23            }
24        }
25        db ans=-1;
26        for(int i=1;i<=n;i++)
27        {
28            db s=f[i][(1<<n)-1]+dis(i,0);
29            if(ans==-1||ans>s) ans=s;
30        }
31        printf("%.2lf\n",ans);
32        return 0;
33    }
```

判断题

1. 第31行输出的ans保留两位小数
2. 若输入 4 1 1 1 -1 -1 1 -1 -1 答案输出7.41
3. 去掉第18行对答案没有影响
4. 若输入的n为0，则答案输出0

选择题

1. 该程序的时间复杂度为（ B ）

A. $O(n^2 logn)$　　B. $O(n^2 2^n)$　　C. $O(n^2)$　　D. $O(n2^n)$

2. 此题f[i][j] 的 [j] 是用来代表什么的（　　）

A. 走过的点　　B. 走过的边　　C. 距离　　D. 以上都错

程序阅读第三题（共 15 分）

给一颗树，带边权，树根是 *S*。每次可以给一条边权+1 并花费 1 的代价，求最小代价使得 *S* 到所有叶子距离相等 每次输入a，b，c代表点a到点b权值为C

树上点的个数 <= 10^6

```
//It's best for everyone to rely on themselves keep moving Lqingyi
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
#define eps 1e-6
#define x first
#define y second
using namespace std;
typedef long long ll;
typedef pair<int,int> PII;
 typedef pair<ll,ll>PLL;
 const int N = 1e6 + 10,M = N * 2;
 int head[N],to[M],last[M],w[M],cnt;
 void add(int a,int b,int c){
     to[++cnt] = b;
     w[cnt] = c;
     last[cnt] = head[a];
     head[a] = cnt;
 }
 ll dis[N],maxn[N],maxn2;
 ll sum;
 int g;
 void dfs(int x,int pre){
     int s = 0;
     for(int i = head[x]; i != -1; i = last[i]){
         int j = to[i];
         if(j == pre) continue;
         s++;
         dis[j] = dis[x] + w[i];
         dfs(j,x);
         maxn[x] = max(maxn[x],maxn[j]);
         maxn2 = max(maxn2,maxn[x]);
     }
     if(s == 0){
         maxn[x] = dis[x];
     }
 }
 void dfs2(int x,int pre,int y){
     if(maxn[x] + y < maxn2){
         sum += maxn2 - maxn[x] - y;
         for(int i = head[x]; i != -1; i = last[i]){
             int j = to[i];
             if(j == pre) continue;
```

```
43                dfs2(j,x,maxn2 - maxn[x] - y + y);
44            }
45        }else{
46            for(int i = head[x]; i != -1; i = last[i]){
47                int j = to[i];
48                if(j == pre) continue;
49                dfs2(j,x,y);
50            }
51        }
52  }
53  int main(){
54     int n;
55      cin >> n;
56      memset(head,-1,sizeof head);
57      int start;
58      cin >> start;
59      for(int i = 1; i <= n - 1; i++){
60          int x,y,z;
61          scanf("%d%d%d",&x,&y,&z);
62          add(x,y,z);
63          add(y,x,z);
64      }
65      dfs(start,0);
66      dfs2(start,0,0);
67      cout << sum << endl;
68      return 0;
69  }
```

判断题

1. 若去掉第23 27 33 34 35行代码对结果没有影响。
2. 若把65行代码改成dfs(start,10000000)对结果没有影响。
3. 若输入的是

```
3
1
1 2 1
1 3 3
```

   答案为2。

4. 12行的M都换成N对结果有影响吗

选择题

1. 若输入

```
6
1
1 2 1
6 1 2
5 4 3
2 4 2
2 3 4
```

   答案为()

A.1 B.2 C.3 D.4

2. 本题的复杂度为()

A.$O(n)$ B.$O(logn)$ C.$O(logn^2)$ D.$O(log^2n)$

3. 此题采用的思想是()

A.分治 B.动态规划 C.贪心 D.以上都不正确

三、程序填空(每题 3 分，共 30 分)

1.(种树)有n个坑相邻的坑之间不能种一棵树，每种一个坑获得一些利益，最多种k棵树，求最大利益 n <= 5e5,k <= n / 2

思路：带反悔的贪心，可以使用模拟链表存下每个坑，用优先队列记录pair ，pair第一个参数是利益，第二个参数是最后一个坑的位置，每次选择利益最大的坑，选择当前坑后需要删除在模拟链表中当前坑的上一个坑和下一个坑，并且把当前坑的价值更改为上一个坑的价值+下一个坑的价值-当前坑的价值. 例如 模拟链表为 1 2 3 4 5 他们的价值分别为 w[1],w[2],w[3],w[4],w[5],选中3种树后，总价值加上w[3]，模拟链表变为 1 3 5 价值变为 w[1],w[2] + w[4] - w[3],w[5]，再选中3种树后总价值加上w[3]就相当于在2,4种树，此时模拟链表变为 3,价值变为w[1] + w[3] + w[5] - w[2] - w[4]，相当于在第2个坑和第3个坑种树。

```
#include<bits/stdc++.h>
#define x first
#define y second
#define endl "\n"
using namespace std;
const int N = 5e5 + 10;
typedef long long ll;
typedef pair<ll,ll> PII;
int n,m;
 ll a[N];
 int flag[N];
 ll to[N],last[N],w[N];
 void addl(int x,int y,ll z){
     to[last[y]] = x;
     last[x] = last[y];
     to[x] = y;
     w[x] = z;
     last[y] = x;
 }
 void addr(int x,int y,ll z){
     last[to[y]] = x;
     to[x] = (*1*);
     to[y] = x;
     w[x] = z;
     last[x] = y;
 }
 void del(int x){
     to[last[x]] = to[x];
     (*2*) = last[x];
     flag[x] = 1;
     to[x] = 0;
     w[x] = 0;
     last[x] = 0;
 }
 priority_queue<PII> q;
 int main(){
```

```
37          cin >> n >> m;
38          for(int i = 1; i <= n; i++){
39              scanf("%lld",&a[i]);
40          }
41          /*if(m * 2 > n){
42              cout << "Error!" <<endl;
43              return 0;
44          }*/
45          for(int i = 1; i <= n; i++){
46              addr(i,i - 1,a[i]);
47              q.push({a[i],i});
48          }
49          last[1] = 0;
50          to[n] = n + 1;
51          w[0] = -0x3f3f3f3f;
52          w[n + 1] = -0x3f3f3f3f;
53          ll sum = 0;
54          ll maxn = 0;
55          for(int i = 1; i <= m; i++){
56              while(flag[q.top().y]) q.pop();
57              PII p = q.top();
58              q.pop();
59              //cout << p.x << " " << p.y << endl;
60              sum += p.x ;
61              maxn = max(maxn,sum);
62              w[p.y] = (*3*);
63              del(last[p.y]);
64              del(to[p.y]);
65              q.push((*4*));
66          }
67          cout << maxn << endl;
68          return 0;
69      }
```

1. 1 处应填

   A. to[y]　　B.y　　C. last[y]　　D. to[last[y]]

2. 2 处应填

   A. last[to[x]]　　B. last[x]　　C. to[x]　　D.x

3. 3 处应填

   A. w[last[p.y]] + w[to[p.y]] - w[p.y]　　B. w[last[p.y]] + w[to[p.y]]　　C. 2 * w[p.y]　　D.w[last[p.y]] - w[p.y]

4. 4 处应填

   A.{maxn,p.y};

   B. {sum,p.y};

   C.{p.x,p.y};

   D. {w[p.y],p.y};

2.

度度熊在玩一个叫做“打怪兽”的游戏。
游戏的规则是这样的。
度度熊一开始会有一个初始的能量值。每次遇到一个怪兽，若度度熊的能量值>=怪兽的能量值并且度度熊剩余血量>=怪兽的攻击力，那么怪兽将会被打败，度度熊的能量值增加1，度度熊的血量减少该怪兽的攻击力，否则度度熊死亡（度度熊的血量刚好减到0时并不会死亡，还能继续战斗），游戏结束。
若怪兽全部打完，游戏也将会结束。
共有n个怪兽，由于度度熊比较弱，它一开始只有1点能量值。
n个怪兽排列随机，也就是说共有n!种可能，度度熊想知道结束时它能量值的期望。
注意这里怪兽的编号是从1开始到编号n为止且编号为i的怪兽能量值为i-1。
由于小数点比较麻烦，所以你只需要输出期望*n!关于1000000007取模后的值就可以了！

在样例中有5个怪兽，它们的能量分别为0,1,2,3,4，其中每个怪兽的攻击力都为1。

收起

CSDN @Lqingyyyy

第一行两个数n,m表示有n只怪兽，度度熊的初始血量(1<=n<=500000,1<=m<=10^9)。接下来一行n个数ai表示编号为i的怪兽的攻击力(0<=ai<=m)。

此题目思路是设 f[i]为至少击败i个怪兽的方案数，那么击败i个怪兽的方案数为 f[i] - f[i + 1] 那么击败怪兽的期望就为 f[1] - f[2] + 2 * (f[2] - f[3]) + ... + n * (f[n] - f[n + 1])答案即为 $\sum_{i=1}^{n} f[i]$
假设sum[i]为1~i的怪兽的攻击力总和。
若sum[i+1]<=m，则f[i]=2^i*(n-i)!。
若sum[i+1]>m，考虑在1~i+1中，去除一个怪兽，使得剩余血量>=0。

```
#include<iostream>
#include<cstring>
#include<queue>
#define x first
#define y second
using namespace std;
const int N = 5e5 + 10,mod = 1e9 + 7;
typedef pair<int,int> PII;
typedef long long ll;
ll a[N];ll b[N],f[N];
ll fac[N];
priority_queue<PII,vector<PII>,(*1*) > q;
ll qpow(ll a,ll b){
    if(b == -1) return 1;
    ll s = 1;
    while(b){
        if(b & 1) s = s * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return s;
}
int main(){
    fac[0] = 1;
    for(int i = 1; i <= 5e5; i++){
        fac[i] = 1ll * fac[i - 1] (*2*) % mod;
```

```
    }
    ll n,m;
    cin >> n >> m;
    for(int i = 1; i <= n; i++){
        scanf("%lld",&a[i]);
        b[i] = b[i - 1] (*3*);
    }
    q.push({a[1],1});
    ll ans = 0;
    ll sum = 0;
    sum += 1;
    for(int i = 1; i <= n - 1; i++){
        q.push({a[i + 1],i + 1});
        sum += (*4*);
        sum %= mod;
        if(b[i + 1] <= m){
            ans += (*5*) % mod;
        }else{
            while(q.size()){
                PII p = q.top();
                if(b[i + 1] - p.x <= m) break;
                sum -= (*6*);
                sum %= mod;
                sum += mod;
                sum %= mod;
                q.pop();
            }
            ans += sum * fac[n - i] % mod;
        }
        ans %= mod;
    }
    if(b[n] <= m) ans += qpow(2,n - 1);
    cout << ((ans) % mod + mod) % mod << endl;
    return 0;
}
```

1. 1 处应填 ()

   A. greater<PII>　　B.less<PII>　　C. greater<int>　　D.less<int>

2. 2 处应填()

   A. + i　　B. * i　　C. - i　　D. * (5e5 - i)

3. 3 处应填()

   A. + a[i]　　B. - a[i]　　C. * a[i]　　D. * a[n - i]

4. 4 处应填()

   A. fac[n - i]　　B. fac[i]　　C. qpow(2,i)　　D. qpow(2,i - 1)

5. 5 处应填()

   A. qpow(2,i - 1) * fac[n - i]　　B. qpow(2,i) * fac[i - 1]　　C. qpow(2 - 1,i) * fac[i - 1]
   D. qpow(2,i) * fac[n - i]

6. 6 处应填()

   A. qpow(2,p.y - 2)　　B. qpow(2,p.y - 1)　　C. qpow(2,p.y)　　D.qpow(2,p.x)